Streaming Data Management and Time Series Analysis: Progetto Finale

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)

Facoltà di Data Science

A.A 2020/21

Simone Tufano, matricola 816984 (s.tufano1@campus.unimib.it)

Abstract

In questo elaborato si andranno ad illustrare i vari procedimenti svolti per calcolare la previsione oraria dell'andamento della serie temporale assegnata per il periodo 01/09/2020 – 31/10/2020.

Introduzione

L'obiettivo di questo progetto è calcolare la previsione a livello orario per i due mesi successivi ai dati disponibili. In particolare, utilizzando tecniche classiche di modellazione per l'analisi di serie storiche, stimare il periodo compreso tra il 01/09/2020 e il 31/10/2020, per un totale di 1464 osservazioni (24*61). Inizialmente si utilizzeranno modelli lineari come gli *ARIMA* e *UCM*, successivamente un approccio di tipo non lineare (attraverso tecniche di Machine Learning) sfruttando le reti neurali ricorrenti (*LSTM* e *GRU*).
Il progetto è stato interamente svolto utilizzando *Google Colab*[1], quindi un notebook *Python* con versione 3.6.9, ad eccezione dei *KNN*, i quali prevedono un'implementazione ottimizzata per la previsione di serie storiche in una libreria presente in *R* (*tsfknn*)[2]. Quindi, solamente per le celle di codice interessate, è stato modificato l'ambiente del notebook.
Le performance dei modelli saranno confrontati attraverso l'errore medio assoluto (*MAE*).

Analisi esplorativa sui dati

Dopo aver importato i dati ed aver verificato la presenza di dati mancanti, è stata svolta un'aggregazione per assicurare la granularità giornaliera del dato. Da questa analisi, è emerso che nei giorni relativi al cambio dell'ora (rispettivamente il 31/03/2019 e 29/03/2020), la terza ora del giorno era mancante. Quindi, si è provveduto a inserire l'osservazione replicando il valore dell'ora precedente.
Successivamente, per consentire l'elaborazione al programma, è stato modificato l' indice del dataframe in modo che facesse riferimento all'ora per ogni riga, trasformandolo nel seguente formato: *'yyyy-mm-dd hh:mm:ss'* attraverso la funzione *to_datetime()*[3]. A questo punto, si procede con un'analisi grafica della serie storica completa (*Figura 1*), per avere un'idea dei dati in questione.

[1] https://colab.research.google.com/notebooks/intro.ipynb
[2] https://cran.r-project.org/web/packages/tsfknn/index.html
[3] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html

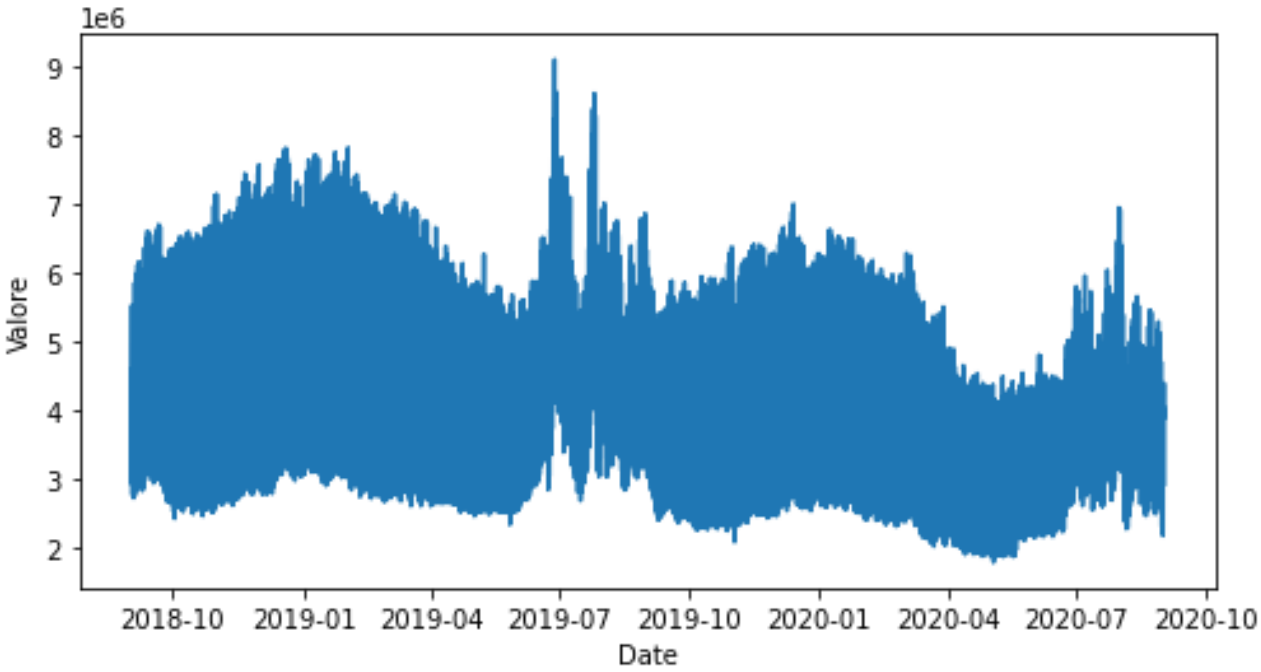


Figura 1. Serie storica iniziale

La serie storica mostra una stagionalità annuale da Ottobre a Giugno, con delle variazioni più alte che sembrano ripetersi nei mesi estivi. Poiché sembrano esserci ulteriori variazioni e stagionalità, si procede con delle visualizzazioni più dettagliate per una maggiore comprensione dei dati.

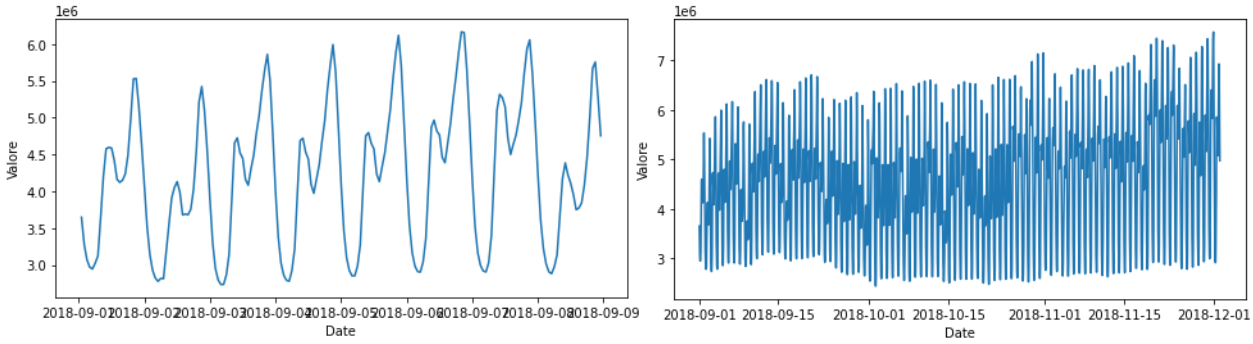


Figura 2. Dettagli giornalieri, settimanali e mensili della serie storica

Dalla *Figura 2*, emerge la conferma che la serie storica mostra un'evidente stagionalità giornaliera e settimanale.

Dopo aver verificato graficamente che i valori della serie ricordano una distribuzione normale[4], si effettuano i test di *Dickey-Fuller* e *KPSS* per valutare se sia necessario eseguire delle trasformazioni o differenziazioni della serie che ne garantiscano la stazionarietà[5]. Poiché entrambi portano al rifiuto dell'ipotesi nulla per tutti i livelli di significatività, non si effettuano trasformazioni della serie.

Quindi, prima di procedere con la divisione del dataset, si procede con un'ulteriore analisi grafica dei correlogrammi (*Figura 3*) della serie storica normale e differenziata di 1 giorno (24 osservazioni). In questo

[4] Appendice, Figura 10

[5] Sezione 'Test di stazionarietà', notebook condiviso.

modo, oltre a confermare la stagionalità (spike rilevanti in 24 e 168), si può avere un'idea per la configurazione dei coefficienti del modello *ARIMA*.

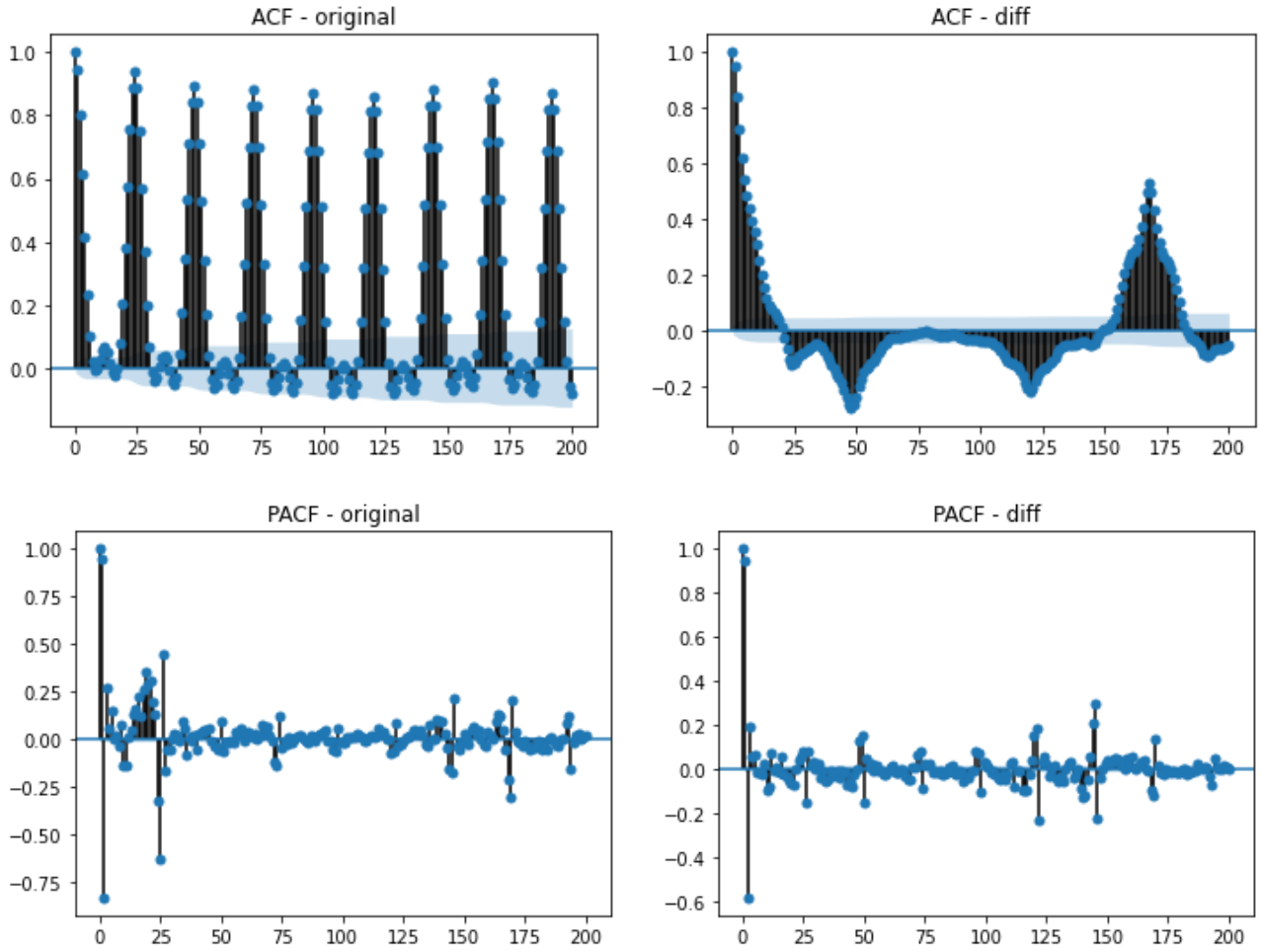


Figura 3. ACF e PACF della serie storica originale e differenziata (24)

Divisione Train e Validation

Il range dell'intero dataset è pari a 1 anno, 11 mesi e 31 giorni, quindi per confrontare l'andamento dei modelli, si sceglie di dividere la serie disponibile in modo che gli ultimi due mesi siano la partizione di validation, ottenendo un train con circa il 90% delle osservazioni.

Modelli Lineari

Lo studio inizia con lo sviluppo di modelli lineari, *ARIMA* e *UCM*.

Il primo modello, suggerito dai correlogrammi, è stato un *ARIMA(3,0,1)(1,1,1)*$_{24}$. Dal grafico delle previsioni sui due mesi finali utilizzati come test, sembra che il modello coglie la stagionalità giornaliera, ma successivamente non riesce ad adattarsi alla stagionalità settimanale[6]. Quindi, attraverso la funzione *auto_arima()*[7], si valuta se attraverso un approccio *Grid Search*[8], l'algoritmo riesce a trovare una

[6] Sezione 'Modelli lineari/Modelli ARIMA/ARIMA(3,0,1)(1,1,1)$_{24}$'
[7] https://pypi.org/project/pyramid-arima/
[8] https://scikit-learn.org/stable/modules/grid_search.html

combinazione di coefficienti con *AIC* minore. Poiché i risultati non migliorano[9], si procede con la creazione di un set di variabili endogene attraverso dei regressori di *Fourier*[10] sia per la stagionalità annuale, sia per la stagionalità settimanale.

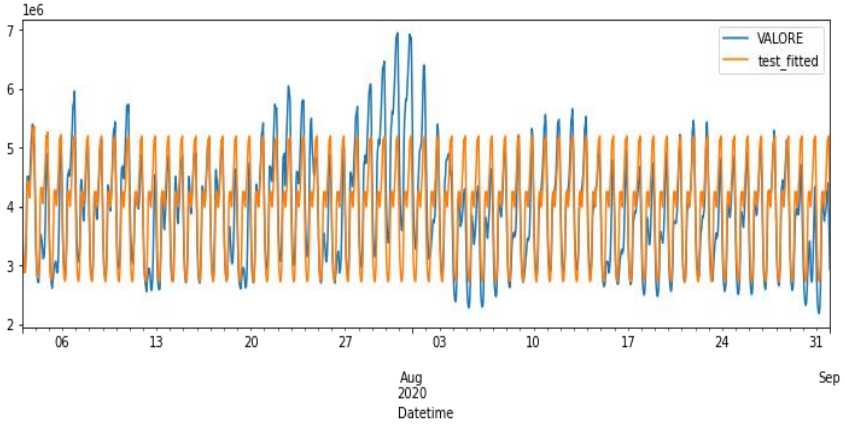


Figura 4. Confronto grafico per modello ARIMA con regressori di Fourier

Nonostante il calcolo di nuovi coefficienti e l'aggiunta dei termini di *Fourier*, il modello sembra comunque non riuscire a cogliere sul lungo termine la stagionalità completa.

Per quanto riguarda i modelli *UCM*, si procede allo stesso modo, ossia utilizzare *Fourier* per correggere la stagionalità annuale e inoltre alleggerire il carico computazionale. Successivamente, dopo diverse strategie inerenti l'inserimento o meno delle componenti, il migliore è risultato il seguente: $Y_t = \mu_t + \gamma_t$, con stagionalità stocastica di tipo dummy con $s_1 = 24$, stagionalità trigonometrica con $s_2 = 168$ (con *20* armoniche), μ_t *Random Walk con Drift*. Di seguito la relativa previsione sul validation.

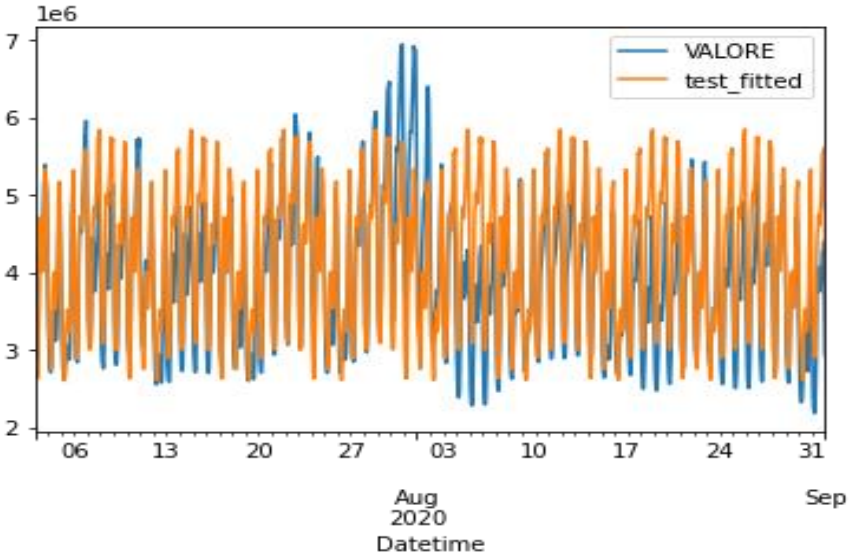


Figura 5. Confronto grafico per modello UCM con regressori di Fourier

Dal grafico segue che il modello a componenti non osservabili riesce a percepire la stagionalità settimanale, modellata correttamente attraverso la forma trigonometrica.

[9] Sezione 'Modelli lineari/Modelli ARIMA/Tentativi con funzione auto_arima()', notebook condiviso.
[10] https://otexts.com/fpp2/complexseasonality.html

Modelli non lineari

Il primo approccio ai modelli non lineare è stato model-free, utilizzando i *KNN*. Come già preannunciato nell'introduzione dell'elaborato, poiché *Python* non presenta una libreria ottimizzata per la costruzione di modelli *KNN* per la modellazione di serie storiche, è stata utilizzata la libreria *tsfknn* in *R*.
Dopo aver importato i dati nel nuovo ambiente, sono stati selezionati i parametri per la previsione ricorsiva, In particolare:

- *t = 16079*
- *p = 1:168*
- *k = (3,5,7,9)*
- *h = 1439*

Quindi, il valore di *k* è stato scelto a seconda della variazione dell' errore medio assoluto sul validation set, che è risultato inferiore per la scelta di *k = 9* (*MAE = 678972.1*). In *Figura 6* il confronto grafico.

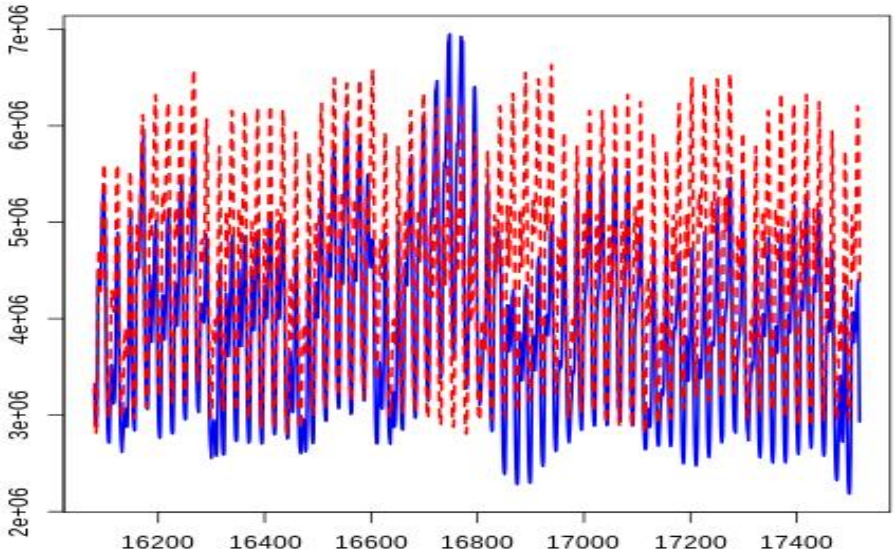


Figura 6. Confronto grafico dei valori previsti (in rosso) e attuali (in blu)

Successivamente, sono state utilizzate delle reti neurali ricorrenti per completare il task. Poiché l'elaborazione di queste reti richiede una struttura diversa del dataset, sono state svolte le seguenti operazioni preliminari:

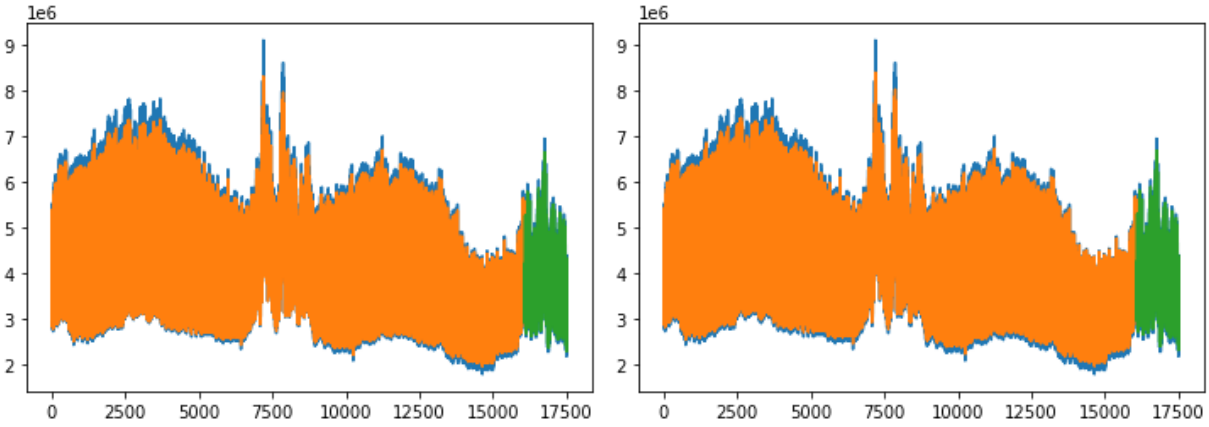
1. Normalizzazione del dataset: funzione MinMaxScaler con range(0,1)[11]*;*

2. Nuova divisione in train e test (con proporzione identica alla precedente)

3. Rimodellazione dei dataset con finestra di lookback (inizialmente 24)

Quindi, i primi modelli lanciati sono stati una *LSTM* e *GRU*, *ognuna rispettivamente con 4 neuroni e attivazione 'Sigmoid', un layer Dense con 1 neurone e attivazione 'Sigmoid'.*
Per la compilazione si è usata come metrica di loss 'mse' e ottimizzatore 'Adam', 10 epoche ed un batch size pari a 1. Dai primi risultati è emerso come la rete con architettura *LSTM* avesse valori di *MAE* inferiori rispetto la rete *GRU*. In *Figura 7* i grafici con le previsioni e le metriche.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

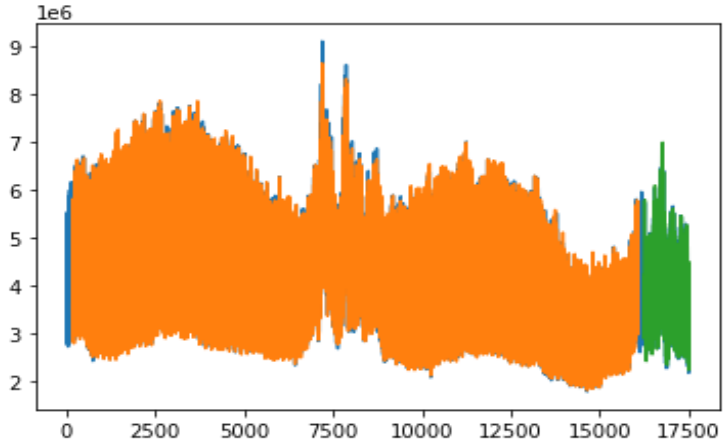


Train: 339868.07, Validation: 236353.38 Train: 340446.64, Validation: 239332.54

Figura 7. Previsioni delle architetture LSTM (sinistra) e GRU (destra), con rispettivi valori di MAE

Quindi, poiché in una prima analisi la *LSTM* si è dimostrata la migliore, si procede per tentativi cambiando il periodo di *lookback (168)*, aumentando layer, unità ed epoche per l'apprendimento. Il modello migliore risulta quindi il seguente:

RNN con layer LSTM, 256 neuroni, layer di output di tipo Dense, 1 neurone, funzioni di attivazione 'sigmoid' (valore di default), addestrata su 15 epoche con batch size pari a 16. Inoltre, per evitare l'overfitting, è stato richiamato il metodo *EarlyStopping()*[12], che monitora la loss del modello con un grado di 'pazienza' pari a 2. In *Figura 8* il grafico con le previsioni.



Train: 91761.41, Validation: 89607.71

Figura 8. Previsioni modello definitivo ML

Tabella con MAE per i migliori modelli (ARIMA, UCM, ML)

Tabella 1. Confronto delle performance dei modelli

	Train_mae	*Validation_mae*
ARIMA	65574.998	423848.513
UCM	98225.747	450642.199
ML	91761.410	89607.710

[12] https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

Previsioni finali e conclusioni

In termini di *MAE*, sul validation set l'approccio *ML* supera di gran lunga i modelli lineari. *UCM* ha un valore maggiore rispetto *ARIMA* nonostante riuscisse a percepire bene l'andamento settimanale (*Figura 5*).
Per la previsione di dati sconosciuti, in assenza di un test set, per *LSTM* è necessario utilizzare uno schema ricorsivo, quindi è stato necessario implementare una funzione che storicizzasse i dati e iterasse il procedimento[13].
Quindi, dopo aver creato gli indici per la creazione del test set e dei regressori di *Fourier* da passare alle funzioni dei modelli lineari e dopo aver ri-addestrato i modelli sulla serie completa, sono state ottenute le seguenti previsioni. (*Figura 9*).

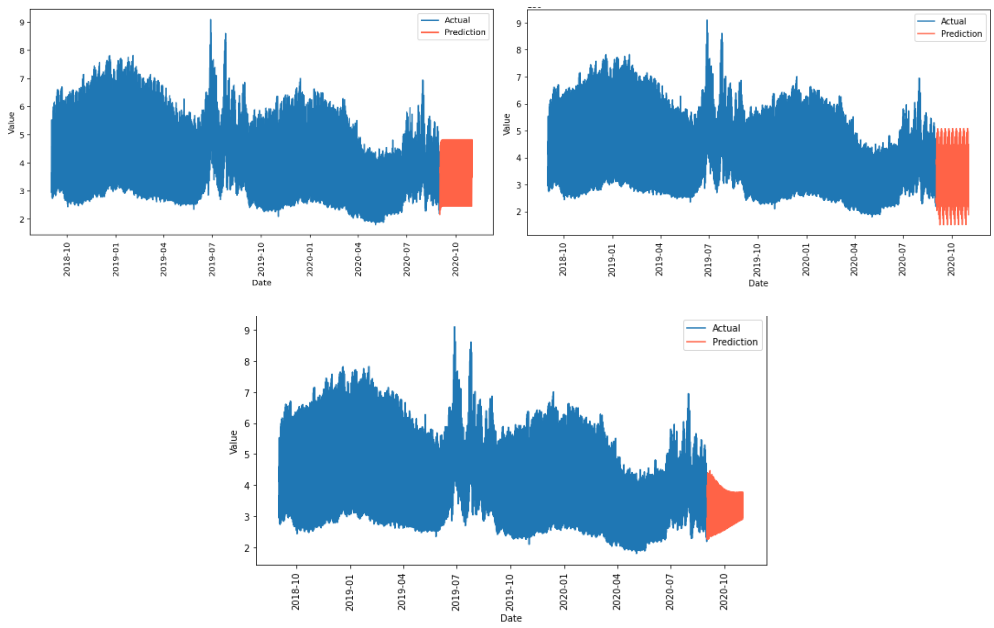


Figura 9. Previsioni dal 01/09/2020 al 31/10/2020 dei modelli ARIMA (in alto a sinistra), UCM (in alto a destra) e LSTM (in basso)

Dai grafici è evidente come *ARIMA* ed *UCM* tendano a ripetersi sul lungo periodo, mentre la rete ricorrente riesce ad identificare autonomamente le stagionalità e il trend e prevedere un andamento della serie per i dati sconosciuti molto più ragionevole con i valori precedenti.
In conclusione, non conoscendo i dati, è possibile ipotizzare che i picchi siano dovuti ad eventi esterni e, aggiungendo delle variabili di intervento che tengano conto di periodi come il trimestre estivo o l'attuale situazione sanitaria dovuta al COVID-19, le performance potrebbero migliorare (almeno per i modelli lineari). Inolte, utilizzando un approccio automatico di tipo Grid Search, aumentando i layer e le unità (non implementato nel codice per cause di peso computazionale), la *LSTM* potrebbe presentare una previsione più attendibile.

[13] Sezione 'Librerie e funzioni', notebook condiviso.

Appendice

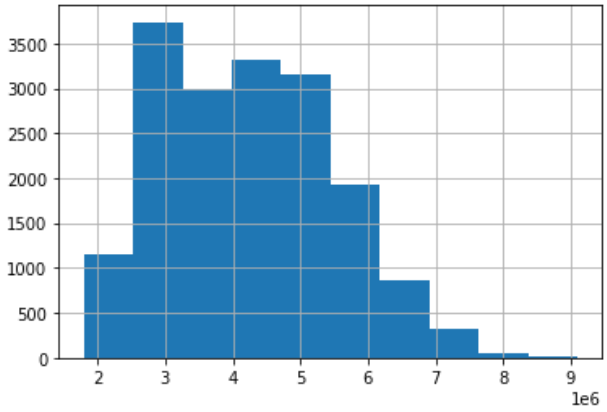


Figura 10. Distribuzione della variabile 'VALORE'

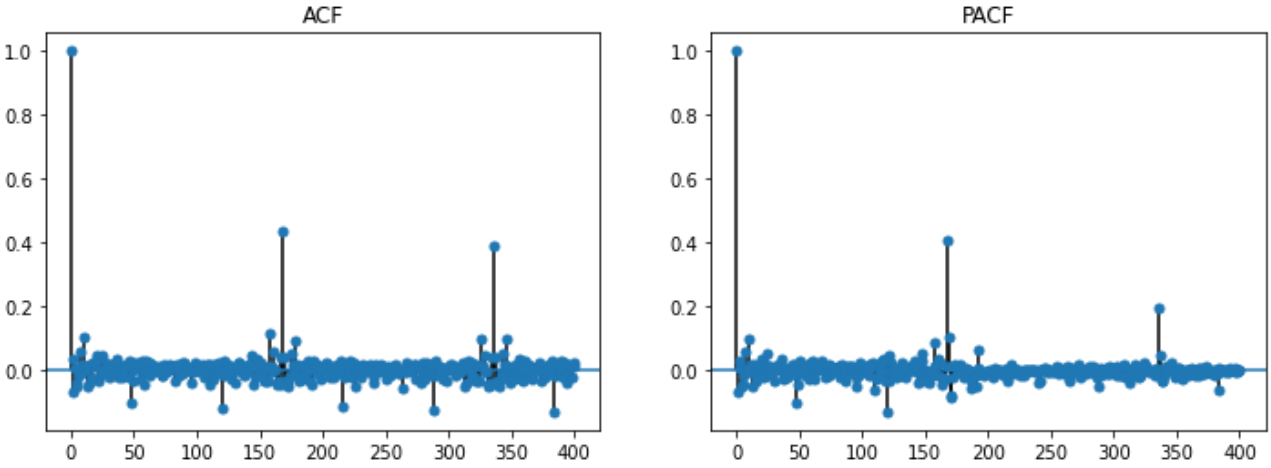


Figura 11. Correlogrammi modello ARIMA definitivo

Tabella 2. Parametri modello ARIMA definitivo

Statespace Model Results

Dep. Variable:	VALORE	No. Observations:	16079			
Model:	SARIMAX(3, 0, 1)x(1, 1, 1, 24)	Log Likelihood	-208261.749			
Date:	Wed, 30 Dec 2020	AIC	416545.498			
Time:	07:36:16	BIC	416630.020			
Sample:	0	HQIC	416573.448			
	- 16079					
Covariance Type:	opg					
	coef	std err	z	P>|z|	[0.025	0.975]
x1	-1.181e-05	1985.637	-5.95e-09	1.000	-3891.776	3891.776
x2	3.851e-05	1666.580	2.31e-08	1.000	-3266.437	3266.438
x3	-5.023e-07	1939.035	-2.59e-10	1.000	-3800.438	3800.438
x4	3.303e-05	1687.182	1.96e-08	1.000	-3306.816	3306.816
ar.L1	2.0535	0.039	52.316	0.000	1.977	2.130
ar.L2	-1.4987	0.057	-26.453	0.000	-1.610	-1.388
ar.L3	0.4185	0.020	20.519	0.000	0.379	0.459
ma.L1	-0.5335	0.039	-13.632	0.000	-0.610	-0.457
ar.S.L24	0.3648	0.005	73.923	0.000	0.355	0.375
ma.S.L24	-0.8661	0.004	-203.711	0.000	-0.874	-0.858
sigma2	1.227e+10	0.031	3.95e+11	0.000	1.23e+10	1.23e+10
Ljung-Box (Q):	698.09	Jarque-Bera (JB):	3548580.04			
Prob(Q):	0.00	Prob(JB):	0.00			
Heteroskedasticity (H):	0.51	Skew:	-1.46			
Prob(H) (two-sided):	0.00	Kurtosis:	75.77			

Tabella 3. Parametri modello UCM definitivo

Unobserved Components Results

Dep. Variable:	VALORE	No. Observations:	16079			
Model:	random walk with drift + stochastic seasonal(24) + stochastic freq_seasonal(168(20))	Log Likelihood	-265453.202			
		AIC	530916.403			
		BIC	530954.809			
Date:	Thu, 31 Dec 2020	HQIC	530929.105			
Time:	09:20:57					
Sample:	0					
	- 16079					
Covariance Type:	opg					
	coef	std err	z	P>|z|	[0.025	0.975]
sigma2.level	5.187e+11	4.22e+13	0.012	0.990	-8.22e+13	8.32e+13
sigma2.seasonal	3.801e+11	8.42e+12	0.045	0.964	-1.61e+13	1.69e+13
sigma2.freq_seasonal_168(20)	3.801e+11	2.59e+12	0.147	0.883	-4.69e+12	5.45e+12
beta.x1	1.526e+08	3.2e+09	0.048	0.962	-6.11e+09	6.42e+09
beta.x2	-1.258e+06	3.7e+08	-0.003	0.997	-7.27e+08	7.24e+08
Ljung-Box (Q):	8499.82	Jarque-Bera (JB):	13768.08			
Prob(Q):	0.00	Prob(JB):	0.00			
Heteroskedasticity (H):	0.58	Skew:	0.20			
Prob(H) (two-sided):	0.00	Kurtosis:	7.52			

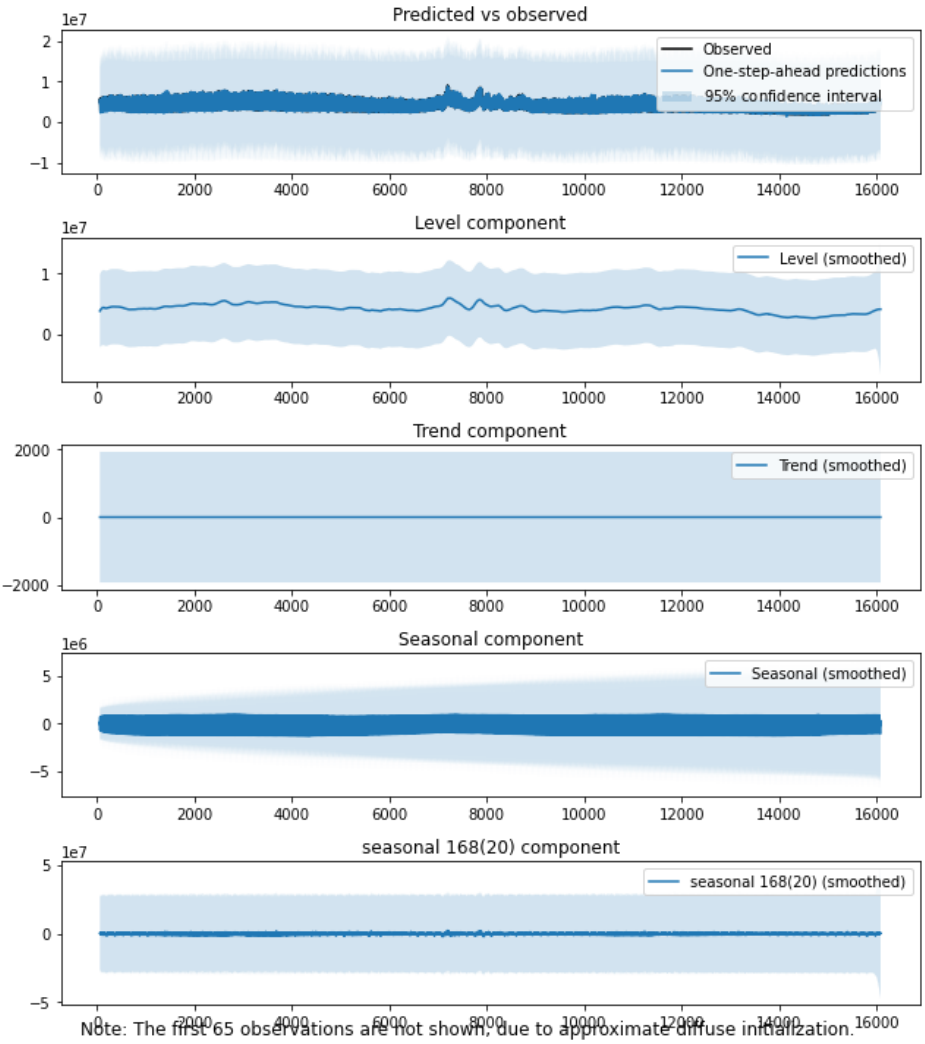


Figura 12. Componenti UCM